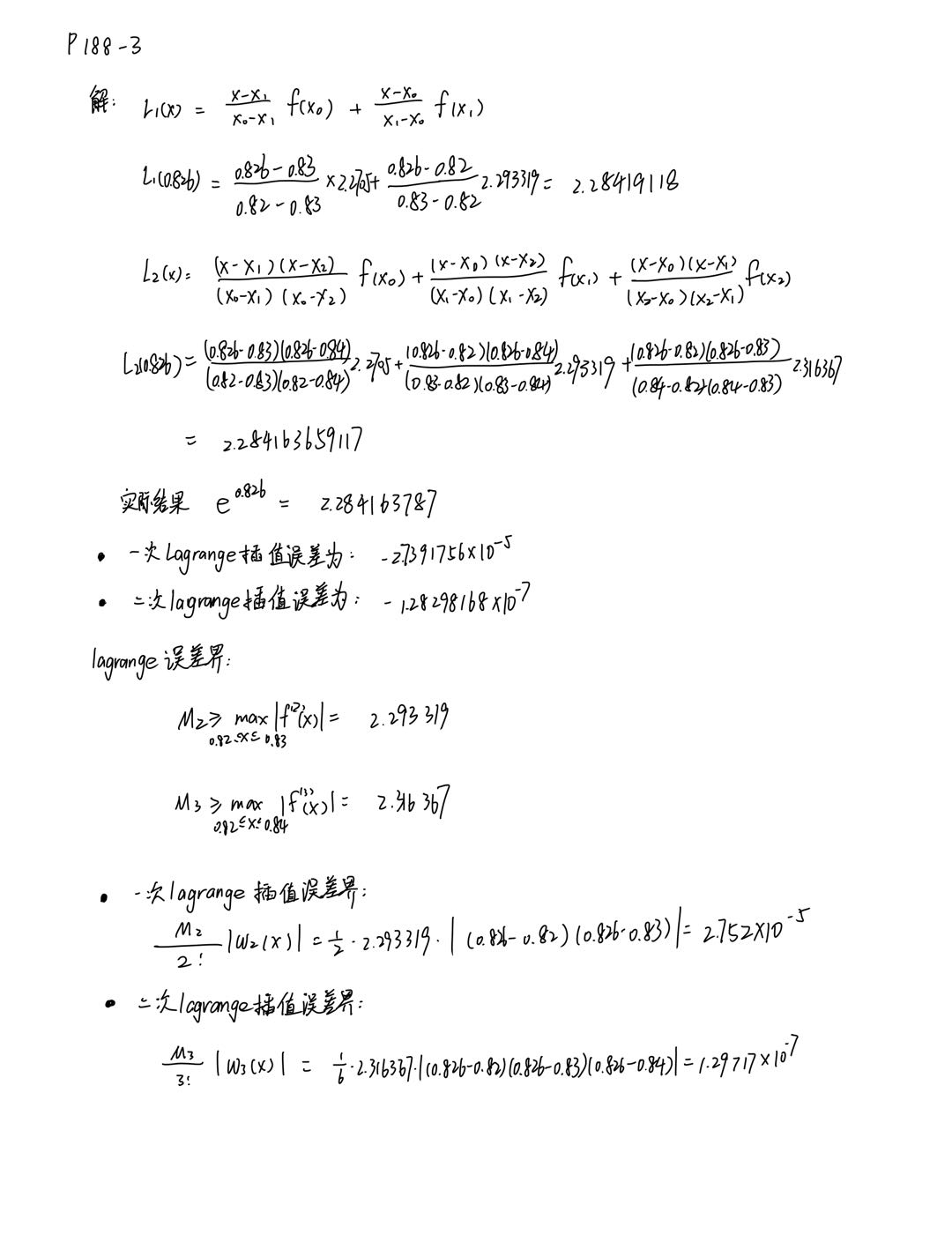
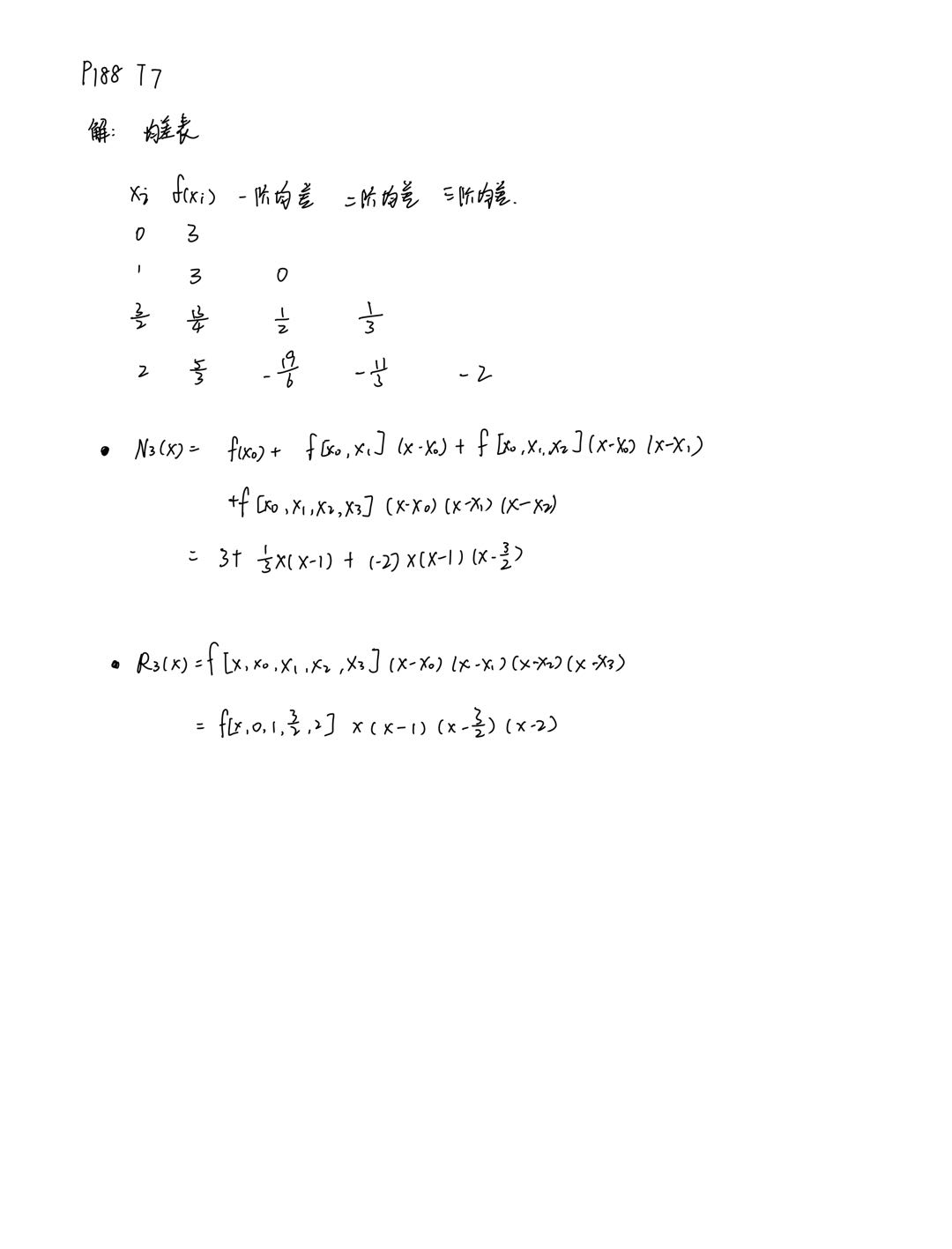
第六次作业

赖显松 2021214726

1 书面作业题目





2 编程题目

2.1 P190 计算实习题T1

**求均差函数：**

钱老师给出的范例，求均差函数（输入：*x*序列，对应的函数值序列f，求解所致最高阶次；输出：从一阶到制定阶次的ndarray list）。代码如下：

1. **def** DivDiff(x,f,order):
2. #给出函数离散节点上的差商/均差(divided differences)
3. #从1阶最高算至order阶
4. #数据点数
5. n = x.size
6. #初始化一个输出list
7. val = []
8. #当前差商为0阶
9. currDD = f.copy()
10. #当前差商长度
11. l = n
12. #开始循环，直到覆盖order阶
13. **for** i **in** range(1,order+1):
14. #自变量差
15. diffx = x[i:n]-x[0:n-i]
16. #函数差，逐阶递推
17. df = currDD[1:l]-currDD[0:l-1]
18. #更新当前差商
19. l = l-1
20. currDD = np.zeros((l,))
21. currDD = df/diffx
22. val.append(currDD)
23. **return** val

其中，三次样条插值用到的均差阶次为二阶，因此在调用这个函数的时候阶次参数输入2。需要将均差输出的list中的二阶元素提取出来使用。

**\*加分项：三对角非线性方程组矩阵求解函数：**

三对角非线性方程组求解LAE函数，输入输出设置与numpy函数库一致：（输入：三对角非线性方程组矩阵A，非线性方程组b；输出：方程组的解x）

代码如下：

1. # 函数：A为三对角矩阵求解LAE,追赶法
2. **def** solveTridiagA(A, b):
3. n = A.shape[0]
4. [L, U] = LUDecomp(A, 'LU')
5. # 求y
6. y = np.zeros((n,))
7. y[0] = b[0]
8. **for** i **in** range(1, n):
9. y[i] = b[i]-L[i, i-1]\*y[i-1]
10. # 求x
11. x = np.zeros((n,))
12. x[n-1] = y[n-1]/U[n-1, n-1]
13. **for** i **in** range(n-2, -1, -1):
14. x[i] = (y[i]-U[i, i+1]\*x[i+1])/U[i, i]
15. **return** x

**三次样条插值函数：**

三次样条插值函数选用了两种边界条件，整体代码较长，分部解释。（输入：待插值的点x\_intp，插值节点序列x，插值节点序列对应的函数值f，边界条件1或2，可选参数传入边界条件）。

**提取边界条件：**

1. # 提取边界条件
2. **if** method == 1:
3. df0, dfn = arg[0], arg[1]
4. **elif** method == 2:
5. d2f0, d2fn = arg[0], arg[1]
6. **else**:
7. **print**('请输入正确的边界条件方法')
8. **return** None

从可选参数arg中取出对应的左右端点一阶导或者是左右端点二阶导。

**统一下标、计算二阶均差：**

1. # 统一n，注意，有n+1个插值节点，但是下表从0到n，这里的n与下标统一
2. n = len(x)-1
4. # 计算所有的二阶均差
5. mean\_df = DivDiff(x, f, 2)
6. mean\_df2 = mean\_df[1]

**计算，和：**

根据公式求解三个序列。

1. #计算h，mu， lamda
2. h = x[1:n+1] - x[0: n]
3. mu = h.copy()
4. **for** j **in** range(1, n):
5. mu[j] = h[j-1]/(h[j-1]+h[j])
6. lamda = -1\*mu + 1

接下来这部分有两种不同的边界条件。

**计算****，构造方程组（边界条件1）：**

1. d = np.zeros(n+1,)
2. d[0] =  6 \* ((f[1]-f[0])/(x[1]-x[0])\*\*2 - df0/(x[1]-x[0]))
3. d[n] =  6 \* (dfn/(x[n]-x[n-1]) - (f[n]-f[n-1])/(x[n]-x[n-1])\*\*2)
4. **for** i **in** range(n-1):
5. d[i+1] = 6 \* mean\_df2[i]
6. A = np.zeros((n+1, n+3))
7. A[0, 1], A[0, 2] = [2, 1]
8. A[n, n], A[n, n+1] = [1, 2]
9. **for** i **in** range(n-1):
10. A[i+1, i+1] = mu[i+1]
11. A[i+1, i+2] = 2
12. A[i+1, i+3] = lamda[i+1]
13. A = A[:, 1:n+2]
14. # 求解M(numpy函数)
15. # M = np.linalg.solve(A, d)
16. # 自己写的三对角追赶法
17. M = lds.solveTridiagA(A, d)

**计算****，构造方程组（边界条件2）：**

1. d = np.zeros(n-1,)
2. **for** i **in** range(n-1):
3. d[i] = 6 \* mean\_df2[i]
4. d[0] -= mu[1]\*d2f0
5. d[n-2] -= lamda[n-1]\*d2fn
6. A = np.zeros((n-1, n+1))
7. **for** i **in** range(n-1):
8. A[i, i] = mu[i+1]
9. A[i, i+1] = 2
10. A[i, i+2] = lamda[i+1]
11. A = A[:, 1:n]
12. # 求解M(numpy函数)
13. # M1 = np.linalg.solve(A, d)
14. # 自己写的三对角追赶法
15. M1 = lds.solveTridiagA(A, d)
16. M = np.concatenate((np.array([d2f0]), M1, np.array([d2fn])))

**求解M（边界条件1）：**

1. # 求解M(numpy函数)
2. # M = np.linalg.solve(A, d)
3. # 自己写的三对角追赶法
4. M = lds.solveTridiagA(A, d)

**求解M（边界条件2）：**

1. # 求解M(numpy函数)
2. # M1 = np.linalg.solve(A, d)
3. # 自己写的三对角追赶法
4. M1 = lds.solveTridiagA(A, d)
5. M = np.concatenate((np.array([d2f0]), M1, np.array([d2fn])))

**求解插值函数值：**

找到插值点在节点序列中的位置并根据插值函数求出插值后的函数值。

1. j = np.searchsorted(x, x\_intp)
2. **if** x\_intp == x[j]:
3. **return** f[j]
4. **else**:
5. j -= 1
6. s1 = M[j]/6/h[j]\*(x[j+1]-x\_intp)\*\*3
7. s2 = M[j+1]/6/h[j]\*(x\_intp-x[j])\*\*3
8. s3 = (f[j]-M[j]\*h[j]\*\*2/6) \* (x[j+1]-x\_intp)/h[j]
9. s4 = (f[j+1]-M[j+1]\*h[j]\*\*2/6) \* (x\_intp-x[j])/h[j]
10. **return** s1 + s2 + s3 + s4

函数准备结束，进入解题部分：

导入数据：

1. # 计算实习题第三题的数据
2. x = np.array([0.9,1.3,1.9,2.1,2.6,3.0,3.9,4.4,4.7,5,6,7,8,9.2,10.5,11.3,11.6,12,12.6,13,13.3])
3. f = np.array([1.3,1.5,1.85,2.1,2.6,2.7,2.4,2.15,2.05,2.1,2.25,2.3,2.25,1.95,1.4,0.9,0.7,0.6,0.5,0.4,0.25])

生成插值序列并用三次样条插值函数求解对应的函数值，将两种边界条件和scipy库中的函数所得到的结果通过subplot显示在同一张图片上。代码如下：

1. xx = np.arange(0.9, 13.3, 0.01)
2. yy = np.zeros(len(xx))
3. **for** i **in** range(len(xx)):
4. yy[i] = itp.cubicSpline(xx[i], x, f, 1, 0, 0)
5. plt.subplot(3,1,1)
6. plt.plot(xx, yy)
7. plt.scatter(x, f)
8. plt.title('边界条件1')

**结果**

三种方法得到的插值结果如图2. 1所示：

可以看出，两种边界条件得到的插值结果在两端会有稍微的不同。当导数值都取0时，边界条件2的插值结果更接近scipy函数库中的插值函数，但是还是有一点不同，scipy的结果在左边的端点处有轻微上凸，1型下凹，2型近似直线。

插值优化的目的是使曲线平滑。考虑端点最近邻的点，将一阶阶均差序列的首尾两侧作为的一阶导边界条件；将二阶均差序列的首尾两侧作为的二阶导边界条件。插值结果如图2. 2所示。

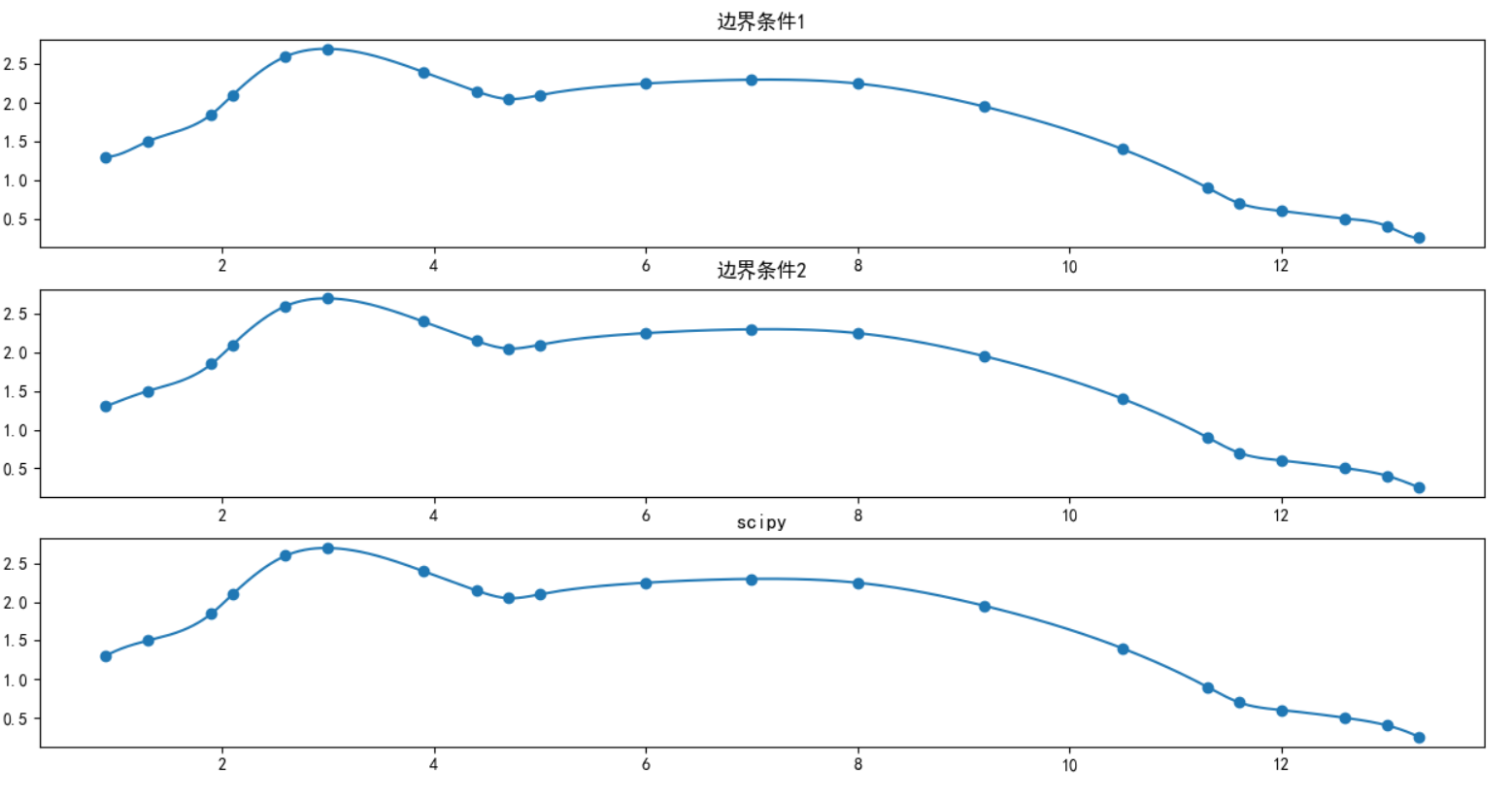


图2. 1 3种方法的插值结果

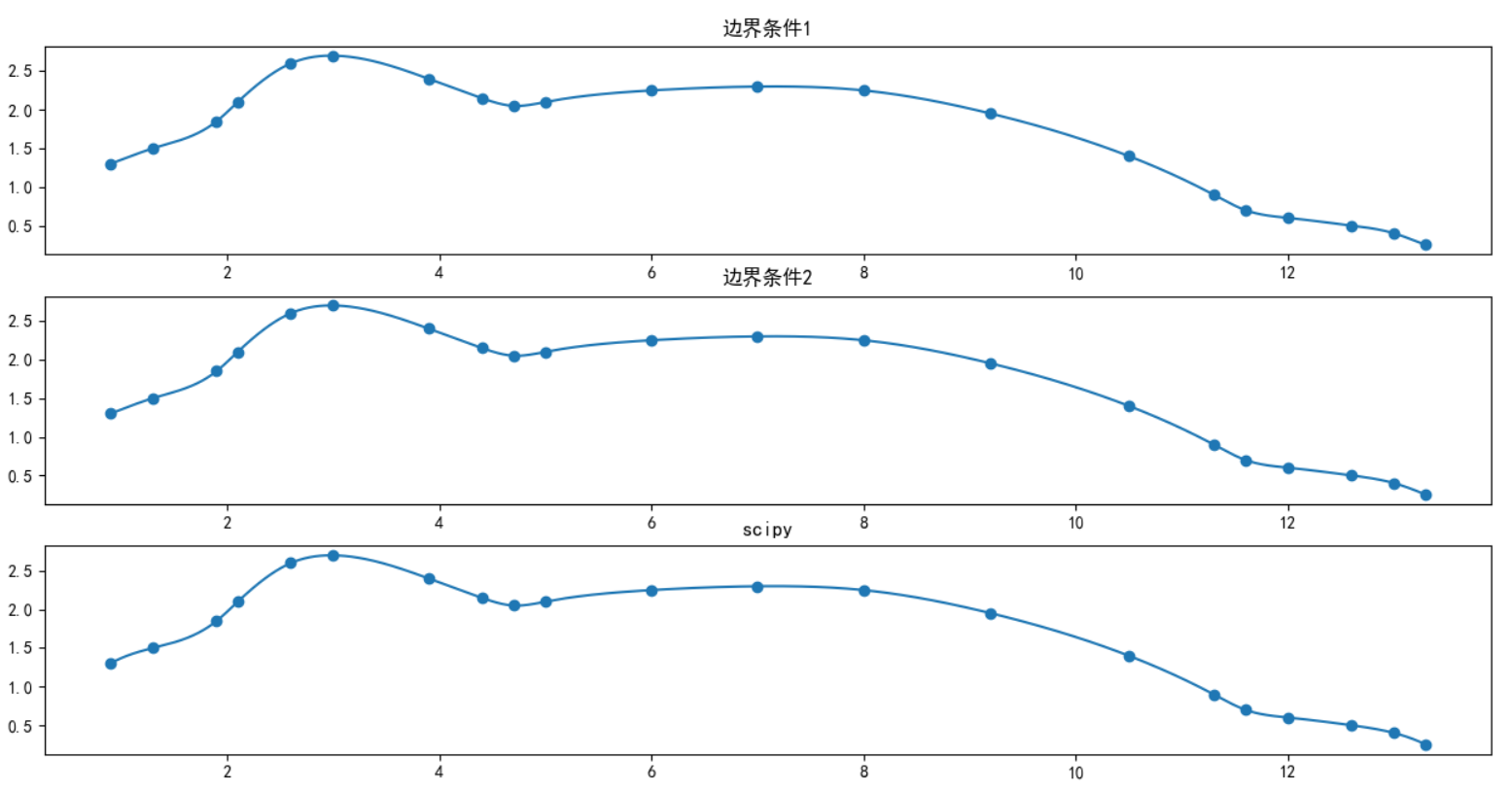


图2. 2 3种方法的插值结果

可以看到，经过边界条件调整的第一种边界条件的三次样条插值结果与第二种边界条件的端点处插值效果很接近了，但是这两种插值与scipy的插值结果还是有一点不同，光滑程度不够。

参考文献

1. 关治．数值方法[M]．北京：清华大学出版社，2006：66-92．

附录

见附件python源代码。